
PRIVACY-PRESERVING DECENTRALIZED FEDERATED LEARNING VIA EXPLAINABLE ADAPTIVE DIFFERENTIAL PRIVACY

Fardin Jalil Piran

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
fardin.jalil_piran@uconn.edu

Zhiling Chen

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
zhiling.chen@uconn.edu

Yang Zhang

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
yang.3.zhang@uconn.edu

Qianyu Zhou

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
qianyu.zhou@uconn.edu

Jiong Tang

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
jiong.tang@uconn.edu

Farhad Imani

School of Mechanical, Aerospace,
and Manufacturing Engineering
University of Connecticut
Storrs, CT 06269
farhad.imani@uconn.edu

December 9, 2025

ABSTRACT

Decentralized Federated Learning (DFL) enables collaborative model training without a central server, but it remains vulnerable to privacy leakage because shared model updates can expose sensitive information through inversion, reconstruction, and membership inference attacks. Differential Privacy (DP) provides formal safeguards, yet existing DP-enabled DFL methods operate as black-boxes that cannot track cumulative noise added across clients and rounds, forcing each participant to inject worst-case perturbations that severely degrade accuracy. We propose PrivateDFL, a new explainable and privacy-preserving framework that addresses this gap by combining HyperDimensional computing (HD) model with a transparent DP noise accountant tailored to decentralized learning. HD offers structured, noise-tolerant high-dimensional representations, while the accountant explicitly tracks cumulative perturbations so each client adds only the minimal incremental noise required to satisfy its (ϵ, δ) budget. This yields significantly tighter and more interpretable privacy-utility tradeoffs than prior DP-DFL approaches. Experiments on MNIST (image), ISOLET (speech), and UCI-HAR (wearable sensor) show that PrivateDFL consistently surpasses centralized DP-SGD and Rényi DP Transformer and deep learning baselines under both IID and non-IID partitions, improving accuracy by up to 24.4% on MNIST, over 80% on ISOLET, and 14.7% on UCI-HAR, while reducing inference latency by up to 76 \times and energy consumption by up to 36 \times . These results position PrivateDFL as an efficient and trustworthy solution for privacy-sensitive pattern recognition applications such as healthcare, finance, human-activity monitoring, and industrial sensing. Future work will extend the accountant to adversarial participation, heterogeneous privacy budgets, and dynamic topologies.

Keywords Decentralized Federated Learning · Privacy-Preserving Machine Learning · Differential Privacy · Explainable Artificial Intelligence · Hyperdimensional Computing

1 Introduction

Federated Learning (FL) has emerged as a powerful paradigm for collaborative model training without the need to centralize raw data, thereby reducing communication overhead and limiting direct exposure of sensitive information such as financial and healthcare records [1, 2]. In its conventional form, Centralized Federated Learning (CFL)

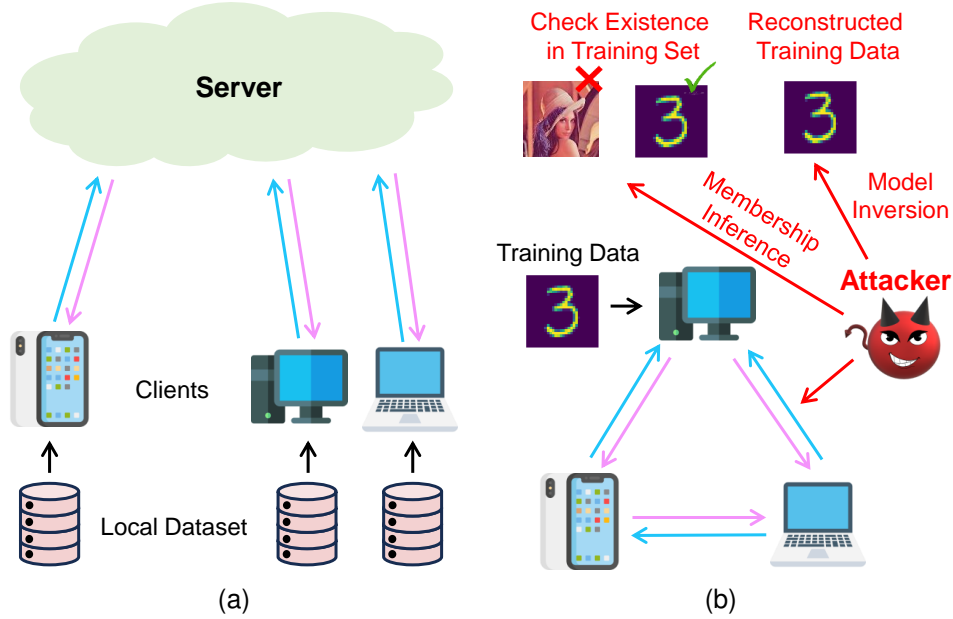


Figure 1: (a) Centralized federated learning framework. (b) Decentralized federated learning setting showing potential data leakage through model inversion and membership inference attacks.

employs a single server to aggregate local model updates from distributed clients, as illustrated in Fig. 1a. Although CFL mitigates data-sharing concerns, the reliance on a central coordinator introduces a single point of failure and heightens susceptibility to privacy and security threats, including model leakage, reconstruction of sensitive features, and gradient-based inference attacks [3].

To overcome these limitations, Decentralized Federated Learning (DFL) removes the central coordinator by enabling direct peer-to-peer communication among clients [4, 5]. This architecture enhances scalability, robustness, and fault tolerance, and is particularly well-suited for distributed pattern recognition tasks in sensor-rich environments such as smart manufacturing, autonomous systems, and Internet of Things (IoT) networks [6–8]. Despite these advantages, DFL remains vulnerable to several sophisticated privacy attacks. Model inversion attacks allow adversaries to reconstruct sensitive features of the underlying training data using intermediate model updates [9]. Membership inference attacks further enable adversaries to determine whether specific data instances contributed to model training, as illustrated in Fig. 1b [10]. These vulnerabilities underscore the need for secure and trustworthy decentralized learning frameworks capable of protecting sensitive information while preserving high model utility.

A variety of privacy-enhancing techniques have been explored to mitigate these vulnerabilities in decentralized learning. Early approaches rely on anonymization, which aims to remove or obscure identifiable attributes in shared model updates. Anonymization alone, however, provides limited protection against modern inference attacks, as adversaries can still reconstruct sensitive information from gradients or output statistics even when explicit identifiers are removed [9]. This weakness becomes especially pronounced in high-dimensional pattern recognition tasks such as image analysis, sensor classification, and multimodal recognition, where subtle statistical correlations can inadvertently leak private information.

More rigorous cryptographic solutions, such as Homomorphic Encryption (HE), enable computation directly on encrypted data and thus offer strong privacy protection [11, 12]. Despite these advantages, HE incurs substantial computational and memory overhead, rendering it impractical for large-scale decentralized learning on resource-constrained clients [13]. Secure Multiparty Computation (SMC) provides another privacy-preserving option by allowing clients to jointly compute functions without revealing their local data. Heavy communication requirements and strict synchronization, however, limit the feasibility of SMC in real-world DFL environments characterized by heterogeneous devices and unreliable network conditions. These challenges emphasize the need for lightweight, scalable, and flexible privacy-preserving mechanisms capable of operating effectively in decentralized, real-time pattern recognition settings.

Differential Privacy (DP) has become a widely adopted mechanism for mitigating privacy risks in DFL. By injecting carefully calibrated noise into model updates, DP reduces the influence of individual data points and provides protection

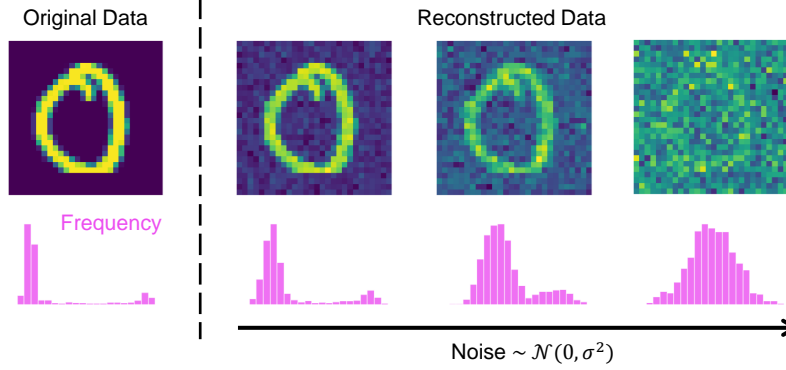


Figure 2: Reconstruction of a differentially protected data point under varying noise levels.

against attacks such as model inversion and membership inference [14–17]. Achieving an appropriate balance between privacy and utility remains challenging in decentralized environments. Excessive noise can severely degrade accuracy, whereas insufficient noise leaves the system exposed to privacy leakage [18–20]. As illustrated in Fig. 2, the level of injected noise directly affects both the strength of privacy guarantees and the resulting model performance. These considerations motivate the development of adaptive, context-aware noise mechanisms that preserve formal privacy guarantees without compromising the accuracy of decentralized pattern recognition models.

Standard DFL operates in a black-box manner, giving clients no visibility into the privacy noise already injected by previous participants. As a result, existing DP-based DFL methods assume worst-case exposure and add a full dose of DP noise at every update, even when substantial noise has already accumulated in the shared model. This redundant noise injection inflates variance, accelerates accuracy degradation, and undermines the ability to maintain effective privacy–utility tradeoffs during long training sequences. The absence of mechanisms for inspecting or verifying how noise evolves across communication rounds further renders privacy guarantees opaque and prevents meaningful auditing in safety-critical pattern recognition applications [21]. These limitations motivate the need for a DFL framework that can track cumulative noise, prevent unnecessary noise amplification, and provide transparent enforcement of privacy budgets throughout collaborative training.

To address these gaps, we introduce Private Decentralized Federated Learning (PrivateDFL), a fully decentralized and explainable DP framework. As illustrated in Fig. 3, PrivateDFL incorporates an eXplainable Artificial Intelligence (XAI)-guided noise accountant that records the cumulative noise injected by all previous clients and communication rounds, computes the required noise needed to satisfy the current DP budget after each local update, and injects only the incremental difference between these two quantities. The accountant is not a post-hoc explainer but a built-in transparency mechanism that makes DP behavior traceable, auditable, and mathematically verifiable across decentralized exchanges. PrivateDFL also integrates a HyperDimensional computing (HD) model whose high-dimensional distributed representations are naturally stable under noise and support efficient and robust updates on resource-constrained devices. Clients are organized in a ring topology and sequentially update a shared HD model using their local data, enabling a fully serverless and communication-efficient training pipeline. Together, these components create a privacy-preserving, interpretable, and accuracy-stable DFL framework suitable for real-time and high-stakes pattern recognition tasks. Our main contributions are summarized as follows.

- We present PrivateDFL, the first DFL framework that unifies DP with an XAI-guided noise accountant. The framework provides transparent and auditable tracking of privacy noise throughout all training rounds.
- We propose an adaptive privacy mechanism that monitors the cumulative noise injected by previous clients, determines the required noise necessary to satisfy the current privacy budget, and injects only the incremental difference. This mechanism avoids worst-case noise injection and substantially improves the privacy–utility tradeoff in fully serverless DFL.
- We integrate an HD model to support efficient and noise-resilient updates while enabling interpretable privacy accounting in high-dimensional feature spaces. This design helps PrivateDFL preserve accuracy under DP noise and remain computationally lightweight for resource-constrained devices.
- We conduct comprehensive experiments on MNIST, ISOLET, and UCI-HAR under Independent and Identically Distributed (IID) and non-IID conditions, benchmarking PrivateDFL against Transformer-based and deep learning models trained with Differential Privacy-Stochastic Gradient Descent (DP-SGD) and Rényi

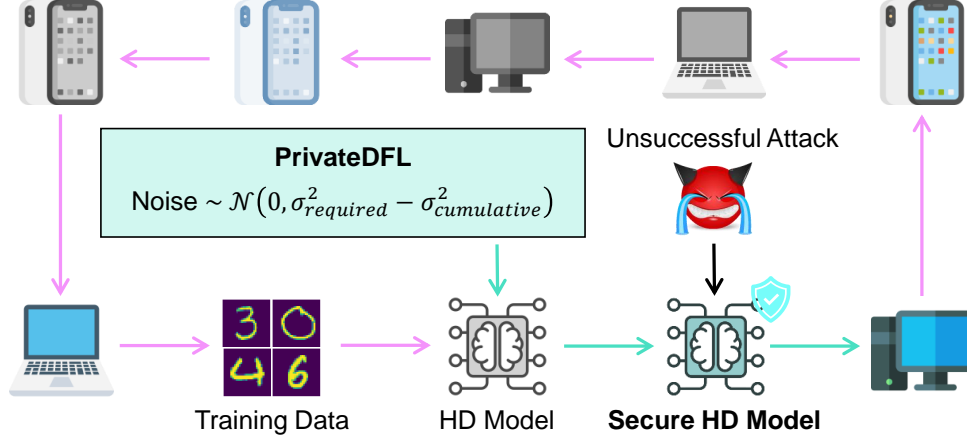


Figure 3: Overview of the PrivateDFL framework with XAI-guided noise accounting and decentralized HD-based model updates.

Differential Privacy (RDP). The results demonstrate significant improvements in accuracy, latency, and energy efficiency while maintaining formal (ϵ, δ) privacy guarantees.

The remainder of this paper is organized as follows. Section 2 reviews prior work on DFL, DP, and HD. Section 3 introduces the fundamentals of HD and DP. Section 4 presents the proposed PrivateDFL framework and its implementation. Section 5 outlines the experimental setup, and Section 6 reports and analyzes the results. Finally, Section 7 summarizes the contributions and highlights future research directions.

2 Research Background

2.1 Decentralized Federated Learning

CFL enables collaborative and privacy-preserving model training but faces several well-documented limitations. CFL must contend with client heterogeneity [22], constrained communication resources [23], and the computational overhead of repeated aggregation [24]. Concerns regarding fairness, reliability, and security further challenge its deployment, because the central server becomes a single point of failure and a primary target for malicious attacks [25–27]. These limitations motivate the development of DFL, which removes the central coordinator and improves robustness by distributing responsibilities across clients.

Several DFL frameworks have been proposed to enable federated training without reliance on a central server. Fully DFL, introduced by Lalitha et al. [28], replaces global aggregation with direct peer-to-peer communication among clients. Decentralized federated averaging, proposed by Lian et al. [29], extends the classic Federated Averaging algorithm to a decentralized topology in which clients exchange and update their models through interactions with local neighbors. Blockchain-based DFL leverages distributed ledger technologies to provide tamper-resistant and secure model aggregation in the absence of a central authority [30]. Graph-based FL formalizes client interactions as a graph structure that supports efficient local aggregation and flexible communication patterns [4]. Collectively, these frameworks employ decentralized optimization strategies that improve scalability, fault tolerance, and resilience in distributed and resource-constrained environments such as IoT networks.

Although DFL removes the central point of failure and reduces dependence on a trusted coordinator, it does not eliminate privacy risks. Clients still exchange intermediate model updates, which remain vulnerable to model inversion and membership inference attacks that can reveal sensitive properties of the training data. These vulnerabilities underscore the need for integrating DP to provide formal privacy guarantees and to minimize data leakage risks. Achieving strong privacy protection while preserving model utility remains a central challenge for DFL, motivating the development of adaptive privacy-preserving mechanisms that dynamically regulate injected noise and enhance security in decentralized learning environments.

2.2 Differential Privacy

DP, introduced by Dwork et al. [31], provides a rigorous framework for safeguarding sensitive data by injecting carefully calibrated noise into computations. The output distribution of a DP mechanism changes only marginally when a single record is added or removed from the dataset, which ensures that individual contributions remain indistinguishable from one another. This property limits the effectiveness of reconstruction and presence testing attacks, including model inversion and membership inference, and has consequently become central to privacy-preserving machine learning. A widely used approach for integrating DP into learning pipelines is DP-SGD, which clips per-sample gradients and injects noise into aggregated gradients during training. When properly configured, DP-SGD offers quantifiable privacy guarantees while preserving useful model performance [32].

RDP extends the classical DP framework by characterizing privacy loss through Rényi divergence [33]. Rather than relying on strict worst-case bounds, RDP provides a more flexible composition mechanism for tracking cumulative privacy loss across multiple iterations. This characteristic aligns naturally with iterative optimization procedures such as SGD, where each step gradually consumes part of the privacy budget. By selecting the order of the divergence and converting RDP guarantees to the standard (ϵ, δ) form, practitioners can calibrate the injected noise more precisely and achieve an improved privacy–utility balance. The additional mathematical complexity of RDP, however, can hinder its practical deployment.

DP has been incorporated into various FL frameworks to strengthen privacy and protect client data. Gong et al. combined DP with HE to protect gradient updates from the server, although complete mitigation of information leakage remains challenging [34]. Wu et al. applied DP to multi-task FL by adding Gaussian noise to model parameters; however, the resulting defense may still be vulnerable to reconstruction attacks and label flipping [35]. Cyffers et al. analyzed a decentralized approach using a relaxed variant of local DP that balances privacy with utility [18]; nevertheless, decentralized settings can heighten exposure to data poisoning and related adversarial threats [36]. Tran et al. introduced a secure decentralized protocol in which the master role rotates among clients to avoid dependence on a fixed central server [37], although collusion among participants can still compromise privacy [36].

Hybrid schemes that combine DP with complementary security mechanisms aim to achieve stronger privacy in decentralized environments. Zhao et al. proposed sharing only partial gradients with Gaussian perturbations and routing updates through an intermediary proxy to obscure client identity [38], although the resulting robustness against inference attacks remains insufficiently characterized. Yin et al. integrated functional encryption with a Bayesian DP formulation to secure communications in federated systems [39], yet reliance on a trusted third party introduces additional risk and may still allow partial reconstruction of sensitive data. Other work adapts DP noise through data-driven encoding strategies to reduce performance degradation while maintaining privacy protection [40]. In cyber-physical systems, DP has been applied through random noise injection to anonymize sensor signals [41], although this approach can impair signal quality. Three-way attribute decision methods for IoT systems structure sensitive features before applying DP, which can improve privacy without inducing substantial accuracy loss [42].

Recent studies highlight the growing importance of XAI in privacy-aware FL. Pairing interpretability techniques with machine learning-based privacy mechanisms improves transparency and supports more informed privacy management. Zheng et al. proposed a dynamic privacy-budget allocation scheme that leverages explanatory signals to estimate workload sensitivity and adjust noise accordingly [43]. The integration of XAI into DP workflows supports more accountable and interpretable privacy control. As decentralized learning continues to expand into high-stakes domains, a key challenge is to design adaptive privacy mechanisms that provide formal and verifiable privacy guarantees while retaining strong model utility.

2.3 Hyperdimensional Computing

HD has gained increasing attention as a lightweight and noise-tolerant alternative to conventional machine learning methods. The high-dimensional representations used in HD enable robust computation under uncertainty and make HD attractive for resource-constrained and distributed learning environments. Recent research has begun exploring how privacy-preserving techniques can be incorporated into HD models to strengthen security and enhance interpretability.

Early work by Hernandez-Cano et al. [44] introduced an intelligent noise-injection strategy that masks less informative features in the original feature space, although the study did not specify the precise noise levels required for formal privacy guarantees. Khaleghi et al. [45] integrated DP into HD workflows to improve security, yet the resulting impact on task accuracy was not fully examined. Piran et al. [46] analyzed the effect of DP on HD models through an explainability perspective and studied how varying noise magnitudes influence performance, but the investigation was limited to standalone HD systems without federated communication. Subsequent work by Piran et al. [47] extended DP to an HD-based CFL framework; however, the reliance on a centralized server for aggregation limited the privacy advantages and did not address decentralized settings.

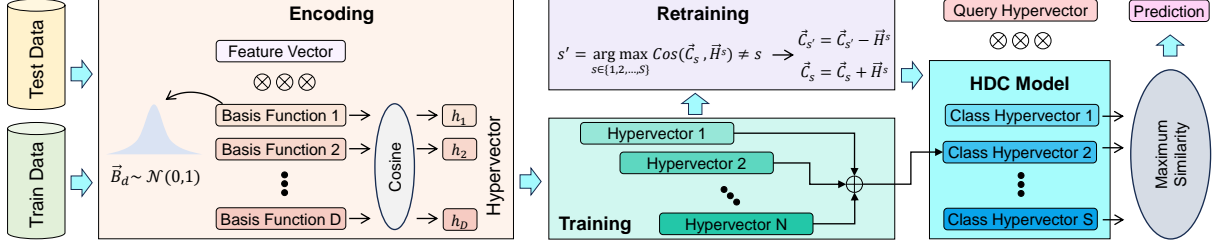


Figure 4: Overview of the hyperdimensional computing model illustrating the encoding, training, inference, and retraining stages.

These studies highlight the need to advance privacy-preserving HD methods that improve both transparency and interpretability while maintaining strong performance in federated environments. Developing adaptive noise-injection mechanisms that preserve the inherent robustness of HD representations and support formal privacy guarantees remains an important direction for enabling trustworthy and effective decentralized learning with HD models.

3 Preliminary

3.1 Hyperdimensional Computing

HD provides an interpretable learning framework based on high-dimensional vector representations known as hypervectors [48–50]. Inspired by brain-like distributed computation [51], HD encodes, aggregates, and compares information through simple vector operations that support robust and explainable pattern recognition [46]. As shown in Fig. 4, HD computation proceeds through four stages: encoding, training, inference, and retraining. These lightweight operations make HD well-suited for decentralized, privacy-preserving learning [47].

Encoding maps an input feature vector \vec{F} into a D -dimensional hypervector \vec{H} using random basis functions $\{\vec{B}_d\}_{d=1}^D$ sampled from $\mathcal{N}(0, 1)$, with components defined as

$$h_d = \cos(\vec{F} \cdot \vec{B}_d). \quad (1)$$

This mapping preserves feature similarity in a structured and interpretable form.

Training constructs class prototypes by summing hypervectors belonging to the same label. For encoded samples \vec{H}^s of class s , the class hypervector is

$$\vec{C}_s = \sum \vec{H}^s. \quad (2)$$

The resulting prototypes capture dominant patterns while remaining stable under noise and variability.

Inference classifies a query hypervector \vec{H}^q by measuring its cosine similarity with each class prototype. The similarity is computed as

$$\text{Cos}(\vec{C}_s, \vec{H}^q) = \frac{\vec{C}_s \cdot \vec{H}^q}{\|\vec{C}_s\| \|\vec{H}^q\|}. \quad (3)$$

The predicted label corresponds to the class with the highest similarity score:

$$q = \arg \max_s \text{Cos}(\vec{C}_s, \vec{H}^q). \quad (4)$$

This similarity-based decision rule provides an interpretable mapping between queries and class prototypes.

Retraining updates class prototypes when misclassifications occur. For a misclassified sample \vec{H}^s predicted as s' , prototypes are corrected by

$$\vec{C}_{s'} = \vec{C}_{s'} - \vec{H}^s, \quad \vec{C}_s = \vec{C}_s + \vec{H}^s. \quad (5)$$

This refinement improves accuracy and maintains stable representations over time.

Overall, HD computing offers interpretable, noise-tolerant, and computationally efficient operations that naturally align with the requirements of decentralized learning.

3.2 Differential Privacy

DP provides a formal guarantee that the inclusion or removal of any single record has only a limited effect on the output of a computation. This protection is achieved by adding calibrated random noise to model updates. The privacy parameters ϵ and δ quantify the strength of this guarantee, where $\epsilon > 0$ denotes the privacy budget that limits how much the output distribution may change when a single record is modified, and $\delta \in [0, 1)$ is a small probability that bounds the likelihood of the guarantee failing.

Definition 1 A randomized mechanism \mathcal{M} satisfies (ϵ, δ) -DP if, for any pair of neighboring datasets I_1 and I_2 differing in one record,

$$\Pr[\mathcal{M}(I_1)] \leq e^\epsilon \Pr[\mathcal{M}(I_2)] + \delta. \quad (6)$$

This condition ensures that an adversary cannot reliably determine whether any specific record is present in the dataset.

Definition 2 The Gaussian Mechanism provides (ϵ, δ) -DP by perturbing the output of a function $g(I)$ with Gaussian noise of variance σ_{dp}^2 :

$$\mathcal{M}(I) = g(I) + \mathcal{N}(0, \sigma_{dp}^2). \quad (7)$$

Definition 3 The sensitivity Δg of a function g is defined as

$$\Delta g = \max_{I_1, I_2} \|g(I_1) - g(I_2)\|, \quad (8)$$

which captures the maximum possible influence of a single record on the output.

Theorem 1 The Gaussian Mechanism satisfies (ϵ, δ) -DP if the noise variance satisfies

$$\sigma_{dp}^2 \geq 2 \frac{\Delta g^2}{\epsilon^2} \ln \left(\frac{1.25}{\delta} \right). \quad (9)$$

4 Research Methodology

4.1 PrivateDFL

PrivateDFL builds on decentralized learning, HD, and DP to provide a secure and interpretable framework for distributed model training. In this formulation, HD offers a noise-tolerant and transparent representational structure for classification, while DFL removes the central server and enables clients to sequentially update and exchange HD models. To protect individual data contributions during these updates, DP is incorporated with parameters (ϵ, δ) , where a smaller ϵ indicates stronger privacy and δ bounds the probability that the guarantee may not hold. Following standard practice, δ is selected to scale inversely with the total number of training samples N , such that $\delta = \delta_0/N$ with $\delta_0 \in (0, 1)$, ensuring that the probability of privacy leakage remains negligible as the dataset grows.

At each communication round r , client k determines its DP noise contribution $\tilde{\Gamma}_k^r$ by computing the difference between the noise required to meet the privacy budget at that point, $\tilde{\xi}_k^r$, and the cumulative noise $\tilde{\Psi}_{k-1}^r$ already present in the received model. This incremental strategy prevents unnecessary amplification of noise and ensures that privacy guarantees are satisfied without degrading model utility.

Training begins with the collaborative construction of the HD model. The first client generates class hypervectors using Equation (2), perturbs them with DP noise as described in Theorem 2, and forwards the secure model to the next participant, with correctness shown in Proof 1. Each subsequent client receives the noisy class hypervectors, constructs its own class hypervectors from local training data, merges them with the received model, and adds additional noise according to Theorem 3, justified in Proof 2.

Theorem 2 The DP noise added to the HD model by the first client in the first round is

$$\tilde{\Gamma}_1^1 \sim \mathcal{N} \left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{1.25N}{\delta_0} \right] \right). \quad (10)$$

Proof 1 The required noise $\tilde{\xi}_1^1$ for the first client is obtained from Equation (9). From Equation (8), the sensitivity of the model is $\Delta g = \sqrt{D}$, since hypervector elements lie in the bounded interval $[-1, 1]$ (attaining ± 1 at the extrema), and the largest possible difference between two neighboring datasets yields an ℓ_2 -norm of \sqrt{D} .

In the worst case, all N training samples for a given client contribute to a single class hypervector. Thus, we set $\delta = \delta_0/N$. Substituting into Equation (9) gives

$$\vec{\xi}_1^1 \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25N}{\delta_0}\right]\right). \quad (11)$$

Since this is the first round with no previously accumulated noise,

$$\vec{\Gamma}_1^1 = \vec{\xi}_1^1. \quad (12)$$

Theorem 3 For any client $k \geq 2$ in the first round, the DP noise added to the HD model satisfies

$$\vec{\Gamma}_k^1 \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{k}{k-1}\right]\right), \quad (13)$$

where k denotes the client index.

Proof 2 From Equation (9) and the sensitivity in Equation (8) with $\Delta g = \sqrt{D}$, the cumulative noise required after client k in round 1 is

$$\vec{\xi}_k^1 \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25kN}{\delta_0}\right]\right), \quad (14)$$

using $\delta = \delta_0/(kN)$ in the worst case. The model received from client $k-1$ already contains

$$\vec{\Psi}_{k-1}^1 \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25(k-1)N}{\delta_0}\right]\right). \quad (15)$$

Choose $\vec{\Gamma}_k^1$ independent of $\vec{\Psi}_{k-1}^1$ so that $\vec{\Psi}_{k-1}^1 + \vec{\Gamma}_k^1$ attains the target variance. Then

$$\begin{aligned} \vec{\Gamma}_k^1 &\sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \left(\ln\left[\frac{1.25kN}{\delta_0}\right] - \ln\left[\frac{1.25(k-1)N}{\delta_0}\right]\right)\right) \\ &= \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{k}{k-1}\right]\right). \end{aligned} \quad (16)$$

In subsequent communication rounds, each client refines the received class hypervectors using Equation (5) with its own private training data. It then calibrates additional DP noise by accounting for the perturbations accumulated in earlier rounds to meet the privacy constraints, and forwards the secure HD model to the next participant. The noise schedule for the first client in these rounds is stated in Theorem 4 and justified in Proof 3, while the schedule for the remaining clients is given in Theorem 5 and justified in Proof 4.

Theorem 4 The DP noise added by the first client in round r after $r-1$ prior communication rounds is

$$\vec{\Gamma}_1^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{K(r-1)+1}{K(r-1)}\right]\right). \quad (17)$$

Proof 3 Let $\vec{\xi}_1^r$ denote the cumulative noise required to protect all contributions present when the first client completes round r , and let $\vec{\Psi}_K^{r-1}$ be the accumulated noise already contained in the model received from client K after $r-1$ rounds. By Equation (8), the sensitivity remains $\Delta g = \sqrt{D}$. In round r , the model aggregates $KN(r-1) + N$ samples, hence

$$\delta = \frac{\delta_0}{KN(r-1) + N}. \quad (18)$$

Substituting into Equation (9) yields

$$\vec{\xi}_1^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25KN(r-1) + 1.25N}{\delta_0}\right]\right). \quad (19)$$

The received model already contains

$$\vec{\Psi}_K^{r-1} \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25KN(r-1)}{\delta_0}\right]\right). \quad (20)$$

Choose $\vec{\Gamma}_1^r$ independent of $\vec{\Psi}_K^{r-1}$ so that their sum attains the target variance. Therefore,

$$\begin{aligned}\vec{\Gamma}_1^r &\sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \left(\ln \left[\frac{1.25 KN(r-1) + 1.25 N}{\delta_0} \right] - \ln \left[\frac{1.25 KN(r-1)}{\delta_0} \right] \right) \right) \\ &= \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{K(r-1) + 1}{K(r-1)} \right] \right).\end{aligned}\quad (21)$$

Theorem 5 For client $k \in \{2, \dots, K\}$ in round r , after $k-1$ clients have updated the class hypervectors, the DP noise added to the HD model is

$$\vec{\Gamma}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{K(r-1) + k}{K(r-1) + k - 1} \right] \right). \quad (22)$$

Proof 4 By Equation (8), the sensitivity is $\Delta g = \sqrt{D}$. When client k completes round r , the model aggregates $KN(r-1) + kN$ samples, so $\delta = \delta_0 / (KN(r-1) + kN)$. Hence

$$\vec{\xi}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{1.25 (K(r-1) + k)N}{\delta_0} \right] \right), \quad (23)$$

and the received model already contains

$$\vec{\Psi}_{k-1}^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{1.25 (K(r-1) + k - 1)N}{\delta_0} \right] \right). \quad (24)$$

Choose $\vec{\Gamma}_k^r$ independent of $\vec{\Psi}_{k-1}^r$ so that their sum attains the target variance. Therefore,

$$\begin{aligned}\vec{\Gamma}_k^r &\sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \left(\ln \left[\frac{1.25 (K(r-1) + k)N}{\delta_0} \right] - \ln \left[\frac{1.25 (K(r-1) + k - 1)N}{\delta_0} \right] \right) \right) \\ &\sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{K(r-1) + k}{K(r-1) + k - 1} \right] \right).\end{aligned}\quad (25)$$

The added noise in each communication round is determined by Theorems 2–5. For the first round, the noise for the initial client is given in Theorem 2 (Equation 10), while the noise for clients $k \geq 2$ is described in Theorem 3 (Equation 13). In subsequent rounds ($r \geq 2$), the noise for the first client is defined in Theorem 4 (Equation 17), and for all other clients in that round it is specified by Theorem 5 (Equation 22).

It is straightforward to observe that Equation 17 can be obtained from Equation 22 by setting $k = 1$. Similarly, Equation 13 is obtained from Equation 22 by substituting $r = 1$. Therefore, the noise injection procedure simplifies to the following rule: the first client in the first round follows Theorem 2, while all other cases, including clients $k \geq 2$ in the first round and all clients in subsequent rounds, follow Theorem 5. The implementation of PrivateDFL based on this unified formulation is presented in Algorithm 1.

4.2 Cumulative Noise

We now derive the total noise accumulated in PrivateDFL after any client k in round r , and compare it with the cumulative noise that arises in a black-box setting where past perturbations cannot be tracked. This comparison highlights why tracking cumulative noise is essential for maintaining accuracy under DP. Without noise accounting, each client must inject the full required noise at every step, leading to rapid and unnecessary variance growth.

4.2.1 Cumulative Noise Under PrivateDFL

Let $\vec{\gamma}_k^r$ denote the cumulative noise in the HD model after client k completes round r . Since PrivateDFL adds only the incremental noise needed to reach the target variance, we have

$$\vec{\gamma}_k^r = \sum_{j=1}^{r-1} \sum_{i=1}^K \vec{\Gamma}_i^j + \sum_{i=1}^k \vec{\Gamma}_i^r. \quad (26)$$

Using the noise schedule in Theorems 2–5 and telescoping of the logarithmic factors, the cumulative noise simplifies to

$$\vec{\gamma}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln \left[\frac{1.25 (K(r-1) + k) N}{\delta_0} \right] \right), \quad (27)$$

Algorithm 1 PrivateDFL: Private Decentralized Federated Learning

```

1: Input  $\{\vec{F}_k\}_{k=1}^K$  ▷ Training samples from  $K$  clients
2: Output  $\{\tilde{C}_{s,K}^R\}_{s=1}^S$  ▷ Final secure HD model after  $R$  rounds (state at client  $K$ )
3: Encoding  $\vec{H} = (\cos(\vec{F} \cdot \vec{B}_d))_{d=1}^D, \{\vec{B}_d\}_{d=1}^D \sim \mathcal{N}(0, 1)$ 
4: ClassHV  $\vec{C}_s = \sum \vec{H}^s, s = 1:S$ 
5: Retrain  $\vec{C}_{s'} \leftarrow \vec{C}_{s'} - \vec{H}^s, \vec{C}_s \leftarrow \vec{C}_s + \vec{H}^s$ 
6: for  $r = 1:R$  do
7:   for  $k = 1:K$  do
8:     if  $r = 1 \wedge k = 1$  then
9:        $\vec{\Gamma}_1^1 \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25N}{\delta_0}\right]\right)$ 
10:       $\{\tilde{C}_{s,1}^1\}_{s=1}^S \leftarrow \text{ClassHV}(\text{Encoding}(\vec{F}_1)) + \vec{\Gamma}_1^1$ 
11:    else if  $r = 1 \wedge k \neq 1$  then
12:       $\vec{\Gamma}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{K(r-1)+k}{K(r-1)+k-1}\right]\right)$ 
13:       $\{\tilde{C}_{s,k}^1\}_{s=1}^S \leftarrow \{\tilde{C}_{s,k-1}^1\}_{s=1}^S + \text{ClassHV}(\text{Encoding}(\vec{F}_k)) + \vec{\Gamma}_k^1$ 
14:    else
15:       $\vec{\Gamma}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{K(r-1)+k}{K(r-1)+k-1}\right]\right)$ 
16:       $\{\tilde{C}_{s,k}^r\}_{s=1}^S \leftarrow \text{Retrain}(\{\tilde{C}_{s,k-1}^r\}_{s=1}^S, \text{Encoding}(\vec{F}_k)) + \vec{\Gamma}_k^r$ 
17:    end if
18:  end for
19:  if  $r < R$  then
20:     $\{\tilde{C}_{s,1}^{r+1}\}_{s=1}^S \leftarrow \{\tilde{C}_{s,K}^r\}_{s=1}^S$ 
21:  end if
22: end for
23: return  $\{\tilde{C}_{s,K}^R\}_{s=1}^S$ 

```

which matches exactly the required noise $\vec{\xi}_k^r$ needed to protect all samples aggregated up to that stage:

$$\vec{\gamma}_k^r = \vec{\xi}_k^r. \quad (28)$$

Setting $r = R$ and $k = K$ gives the final cumulative noise of the trained model:

$$\vec{\gamma}_K^R \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25KR N}{\delta_0}\right]\right). \quad (29)$$

This shows that PrivateDFL exhibits only logarithmic growth in variance with respect to the number of clients and rounds.

4.2.2 Cumulative Noise in the Black-Box Case

In a system where noise cannot be tracked, each client must inject the full required DP noise at every round. Let $\vec{\Xi}_k^r$ denote the cumulative noise under this black-box assumption:

$$\vec{\Xi}_k^r = \sum_{j=1}^r \sum_{i=1}^k \vec{\xi}_i^j. \quad (30)$$

From Proof 4,

$$\vec{\xi}_i^j \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\frac{1.25N j i}{\delta_0}\right]\right), \quad (31)$$

and summing all contributions gives

$$\tilde{\Xi}_k^r \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \sum_{t=1}^{K(r-1)+k} \ln\left(\frac{1.25Nt}{\delta_0}\right)\right). \quad (32)$$

Rewriting the sum as a logarithm of a product yields

$$\begin{aligned} \tilde{\Xi}_k^r &\sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\prod_{t=1}^{K(r-1)+k} \frac{1.25Nt}{\delta_0}\right]\right) \\ &= \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\left(\frac{1.25N}{\delta_0}\right)^{K(r-1)+k}\right] + \frac{2D}{\epsilon^2} \ln[(K(r-1)+k)!]\right). \end{aligned} \quad (33)$$

For the final model ($k = K, r = R$), we obtain

$$\tilde{\Xi}_K^R \sim \mathcal{N}\left(0, \frac{2D}{\epsilon^2} \ln\left[\left(\frac{1.25N}{\delta_0}\right)^{KR} (KR)!\right]\right), \quad (34)$$

which grows super-exponentially due to the factorial term $(KR)!$.

4.2.3 PrivateDFL vs. Black-Box Noise Accumulation

PrivateDFL achieves the minimum DP noise required:

$$\sigma_{\text{PrivateDFL}}^2 \propto \ln(KR), \quad (35)$$

whereas the black-box approach suffers from explosive variance:

$$\sigma_{\text{Black-Box}}^2 \propto \ln[(KR)!]. \quad (36)$$

Thus, cumulative noise tracking is essential for preserving accuracy in decentralized differentially private learning.

5 Experimental Design

We evaluate PrivateDFL on three benchmark datasets spanning distinct sensing modalities: MNIST (image), ISOLET (speech), and UCI-HAR (wearable sensors). This selection enables testing across heterogeneous input domains. Both IID and non-IID settings are considered. In IID, samples are evenly split so each client observes the same data distribution. In non-IID, each client receives only two classes (e.g., MNIST clients may hold $\{0,1\}$, $\{2,3\}$, etc.), reflecting realistic deployments where clients collect skewed data.

The evaluation proceeds in three steps. First, we conduct a sensitivity analysis to study how the privacy budget ϵ , privacy-loss parameter δ_0 , hypervector dimensionality D , the number of clients K , and the number of training samples per client affect accuracy under both IID and non-IID partitions. For IID, we also vary K while keeping the total dataset fixed to isolate the effect of data fragmentation.

Second, we benchmark PrivateDFL against state-of-the-art Transformer and deep learning models trained with DP in centralized settings. These baselines access the full dataset directly and therefore represent an upper bound on accuracy and latency without federated constraints. We implement DP-SGD and RDP using Opacus [52]. MNIST baselines include Vision Transformer, ResNet50, GoogLeNet, and AlexNet; ISOLET baselines include Transformer, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Temporal Convolutional Network (TCN); and UCI-HAR baselines include TabTransformer, TabNet, Deep Neural Network (DNN), and Deep and Cross Network (DCN).

All comparisons are reported in terms of accuracy, training time, inference latency, and energy consumption, using a fixed privacy budget of $\epsilon = 0.4$ and privacy-loss coefficient $\delta_0 = 0.001$.

6 Experimental Results

6.1 Sensitivity Analysis

In this subsection, we examine how key hyperparameters influence the performance of PrivateDFL. The parameters of interest include the privacy budget ϵ , the privacy loss coefficient δ_0 , the number of clients K , the hypervector

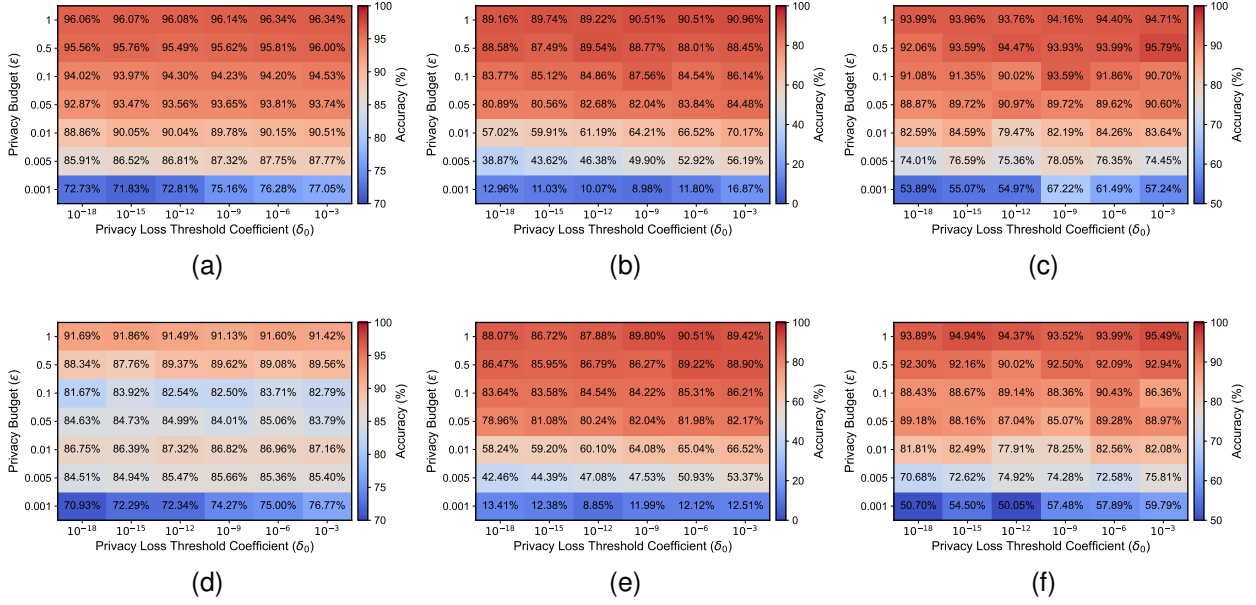


Figure 5: Accuracy of PrivateDFL under varying privacy parameters, including the privacy budget (ϵ) and privacy loss coefficient (δ_0), for both IID and non-IID data distributions. Panels (a)–(c) report results on MNIST, ISOLET, and UCI-HAR under IID partitioning, while panels (d)–(f) show the corresponding outcomes under non-IID splits.

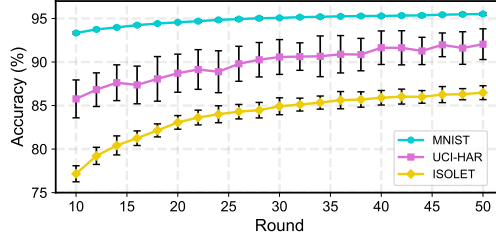


Figure 6: Accuracy of PrivateDFL across communication rounds under IID partitioning. For each dataset (MNIST, ISOLET, UCI-HAR), the curve reports the mean accuracy over client counts $K \in \{10, \dots, 1000\}$, and the shaded region denotes the corresponding ± 1 standard deviation.

dimensionality D , and the number of training samples per client. We begin by analyzing the effect of the privacy parameters under IID conditions.

Figure 5a reports the MNIST accuracy when the dataset is evenly partitioned across 100 clients. With a relatively large privacy budget (e.g., $\epsilon = 1$ and $\delta_0 = 10^{-3}$), the injected noise is minimal and PrivateDFL attains an accuracy of 96.34%. Fixing $\delta_0 = 10^{-9}$ and progressively decreasing ϵ increases the noise magnitude and leads to a monotonic reduction in accuracy: the performance drops from 96.14% at $\epsilon = 1$ to 95.62%, 94.23%, 93.65%, 89.78%, 87.32%, and 75.16% for $\epsilon \in \{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$.

In contrast, varying δ_0 produces only marginal differences. At $\epsilon = 0.1$, tightening δ_0 from 10^{-3} to 10^{-18} results in a change of approximately 0.56 percentage points (from 94.53% to 93.97%). Minor fluctuations also arise from randomness in Gaussian perturbations; for example, at $\epsilon = 0.5$, accuracy varies between 95.49% ($\delta_0 = 10^{-12}$), 95.62% ($\delta_0 = 10^{-9}$), and 95.76% ($\delta_0 = 10^{-15}$), with a maximum difference of only 0.27 percentage points. Similar patterns are observed for ISOLET and UCI-HAR under IID partitions with 100 clients, as shown in Figures 5b and 5c.

Figures 5d, 5e, and 5f extend this analysis to the non-IID setting. The overall trends remain consistent, although the impact of privacy parameters becomes more pronounced for MNIST due to greater heterogeneity across clients. For instance, at $\epsilon = 0.5$ and $\delta_0 = 10^{-6}$, MNIST accuracy declines from 95.81% in the IID case to 89.08% under non-IID conditions. ISOLET and UCI-HAR exhibit smaller discrepancies under the same parameters: 88.01% versus 89.22% for ISOLET and 93.99% versus 92.09% for UCI-HAR.

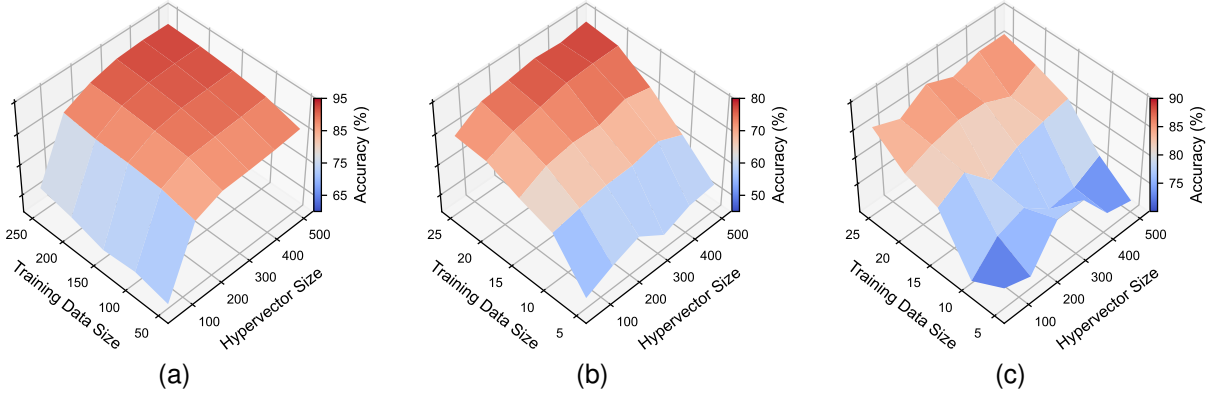


Figure 7: Effect of hypervector dimensionality and the number of training samples per client on model accuracy under IID partitioning. Panels (a)–(c) report results for MNIST, ISOLET, and UCI-HAR, respectively.

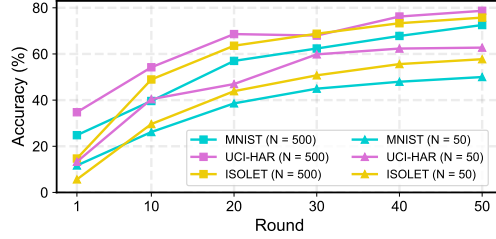


Figure 8: Accuracy of PrivateDFL across communication rounds under non-IID partitioning, shown for varying numbers of training samples per client.

Figure 6 further examines the effect of client count by reporting accuracy as a function of the number of clients under IID partitioning for MNIST, ISOLET, and UCI-HAR across multiple communication rounds. The number of clients K is varied from 10 to 1000 while keeping the total training set fixed and evenly distributed, ensuring that performance changes arise from partitioning and coordination effects rather than differences in data volume. For each round, the figure displays the mean accuracy along with the corresponding standard deviation across client counts. The variability remains small for MNIST (below 1.5%), moderate for ISOLET (up to 5.31%), and larger for UCI-HAR (up to 7.37%). The standard deviation stays nearly constant across rounds for each dataset, suggesting that the primary source of uncertainty is the intrinsic difficulty of the task rather than the progression of communication rounds.

Complementing the client-count analysis in Figure 6, Figure 7 illustrates the joint effect of hypervector dimensionality and the number of training samples per client on model accuracy. Figures 7a, 7b, and 7c correspond to MNIST, ISOLET, and UCI-HAR, respectively. For each dataset, the hypervector dimension is varied from 50 to 500, while the number of training samples per client ranges from 50 to 250 for MNIST and from 5 to 25 for ISOLET and UCI-HAR.

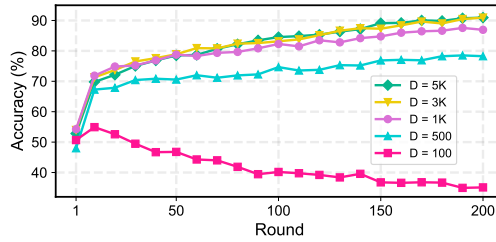


Figure 9: PrivateDFL accuracy across communication rounds for varying hypervector dimensionalities (D) under non-IID MNIST partitioning.

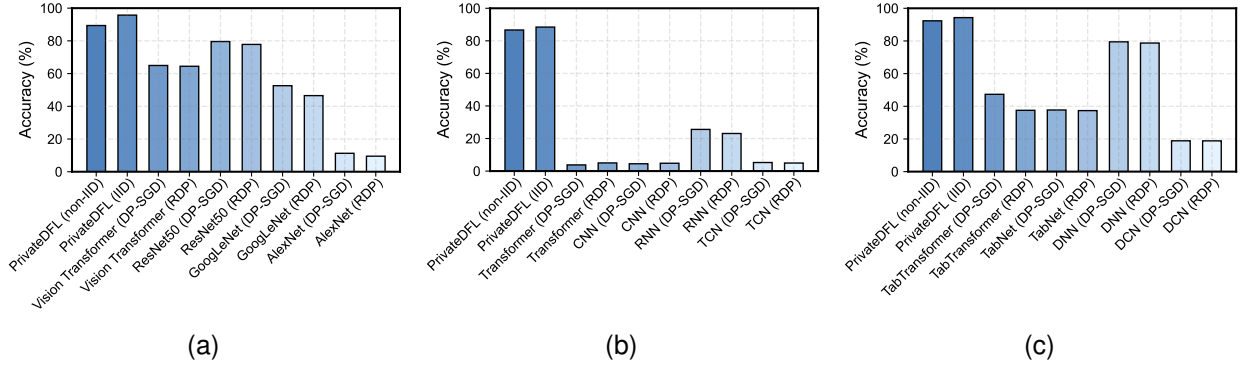


Figure 10: Accuracy comparison between PrivateDFL and differentially private baseline models across the three benchmark datasets: (a) MNIST, (b) ISOLET, and (c) UCI-HAR.

For MNIST, accuracy is highly sensitive to hypervector dimensionality. With 200 training samples per client, accuracy increases from 65.35% at $D = 50$ to 91.86% at $D = 500$. Conversely, when the hypervector size is fixed at $D = 300$, accuracy rises more gradually from 86.68% to 92.27% as the number of samples increases from 50 to 250. These results suggest that hypervector dimensionality plays a more dominant role than training data quantity for MNIST.

In contrast, ISOLET exhibits the opposite trend: accuracy is driven primarily by the number of training samples. With hypervector size fixed at $D = 500$, accuracy improves substantially from 53.56% to 77.99% as the number of samples grows from 5 to 25. However, for a fixed sample count, increasing the hypervector dimension yields only modest gains. At 5 samples per client, accuracy rises by 5.70% as D increases from 50 to 500, and by 8.53% when clients hold 25 samples.

The UCI-HAR dataset follows a similar pattern to ISOLET. When $D = 50$, accuracy increases by 12.04% as the number of training samples rises from 5 to 25; for $D = 500$, the improvement reaches 14.86%. Once clients already possess 25 samples each, further increases in hypervector dimensionality yield only a minor enhancement of 1.05%. These findings indicate that, for ISOLET and UCI-HAR, additional training samples contribute more effectively to performance than expanding hypervector dimensionality.

Extending the analysis to the non-IID setting, Figure 8 reports accuracy across communication rounds while varying the number of training samples per client. For MNIST with 50 samples per client, accuracy begins at 11.68% in round 1 and reaches 50.00% by round 50. Increasing the number of samples consistently improves performance at all rounds; for instance, with more training data, the accuracy in round 1 increases to 24.81%, and the round-50 accuracy reaches 72.53%. Similar trends appear for ISOLET and UCI-HAR. After 50 rounds, accuracy is 57.72% and 62.74% with 50 samples per client, rising to 75.75% and 78.69% when each client has 500 samples. Across all considered hypervector sizes, additional training samples have the strongest impact, yielding consistent accuracy improvements under non-IID partitions.

Finally, Figure 9 evaluates the influence of hypervector dimensionality under non-IID partitions for MNIST. At round 1, accuracy is approximately 50% across all tested dimensions. When the dimension is small (e.g., $D = 100$), the model lacks sufficient representational capacity and the DP noise dominates, causing accuracy to decline over rounds: performance drops to 46.78%, 40.18%, 36.76%, and 35.10% at rounds 50, 100, 150, and 200, respectively. Increasing the dimensionality mitigates this effect. With $D = 500$, accuracy improves progressively with training, reaching 70.57%, 74.66%, 76.86%, and 78.25% over the same rounds. Larger dimensions yield further gains; after 200 rounds, accuracy rises to 86.93% for $D = 1000$ and 90.95% for $D = 3000$, corresponding to improvements of 8.68% and 12.70% over $D = 500$. Beyond this range, the benefit saturates because increased dimensionality also amplifies the effective DP noise. For example, the round-200 accuracy for $D = 5000$ is 90.94%, nearly identical to that for $D = 3000$.

6.2 Benchmark

Figure 10 compares PrivateDFL with differentially private machine learning baselines across MNIST, ISOLET, and UCI-HAR under both IID and non-IID partitions. Across all datasets and distributions, PrivateDFL consistently achieves the highest accuracy while satisfying the same privacy budgets.

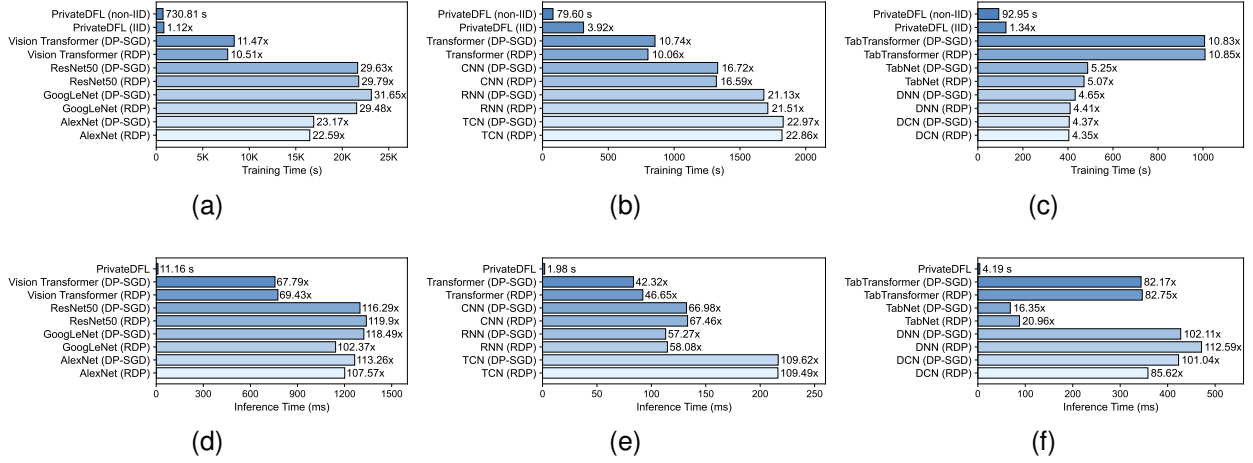


Figure 11: Comparison of runtime performance for PrivateDFL and DP baseline models. The top row shows training time and the bottom row shows inference latency for (a,d) MNIST, (b,e) ISOLET, and (c,f) UCI-HAR.

For MNIST (Figure 10a), PrivateDFL attains 95.74% under IID and 89.38% under non-IID settings. Deep models trained with DP perform substantially worse. With DP-SGD and RDP, AlexNet reaches 11.27% and 9.51%, GoogLeNet achieves 52.60% and 46.53%, ResNet50 reaches 79.59% and 77.83%, and the Vision Transformer attains 64.96% and 64.47%, respectively. These results highlight the robustness of the hyperdimensional backbone in PrivateDFL, which preserves utility more effectively under privacy noise.

For ISOLET (Figure 10b), PrivateDFL again leads with 88.45% accuracy in IID and 86.66% in non-IID conditions. Transformer and CNN baselines trained with DP remain between 3.78% and 5.26%. The RNN baseline reaches 25.59% with DP-SGD and 23.09% with RDP, while the TCN remains below 6%. These wide gaps indicate that sequential deep models are especially sensitive to DP noise, whereas PrivateDFL maintains strong performance.

For UCI-HAR (Figure 10c), PrivateDFL achieves 94.30% in IID and 92.33% in non-IID settings. Among tabular baselines, TabTransformer attains 47.34% with DP-SGD and 37.53% with RDP, TabNet achieves 37.73% and 37.36%, the DNN baseline reaches 79.47% and 78.72%, and DCN yields 18.87% and 18.83%. Across image, speech, and sensor modalities, the results demonstrate that PrivateDFL sustains significantly higher accuracy while adhering to the same privacy guarantees, offering a more favorable privacy-utility balance than conventional DP-trained models.

Building on the accuracy gains shown in Figure 10, Figure 11 compares training time and inference latency for PrivateDFL and differentially private baselines across MNIST, ISOLET, and UCI-HAR under both IID and non-IID partitions. The results demonstrate that PrivateDFL achieves higher accuracy while requiring substantially less computation.

For MNIST (Figures 11a and 11d), PrivateDFL completes training in 820.97 seconds for IID and 730.81 seconds for non-IID. Deep baselines are dramatically slower: GoogLeNet requires 23127.31 seconds with DP-SGD and 21545.07 seconds with RDP; ResNet50 requires 21656.82 and 21768.85 seconds; AlexNet exceeds 16000 seconds; and Vision Transformer requires between 7684.12 and 8384.22 seconds. At inference, PrivateDFL reaches 11.16 milliseconds, whereas the deep models range between 756 and 1338 milliseconds.

For ISOLET (Figures 11b and 11e), PrivateDFL completes training in 311.71 seconds for IID and 79.60 seconds for non-IID. RNN baselines with DP require more than 1680 seconds, TCN exceeds 1820 seconds, and Transformer exceeds 800 seconds. Inference latency further diverges: PrivateDFL requires only 1.98 milliseconds, whereas sequential baselines require between 83 and 216 milliseconds.

For UCI-HAR (Figures 11c and 11f), PrivateDFL trains in 124.85 seconds for IID and 92.95 seconds for non-IID. TabTransformer requires approximately 1007 seconds, TabNet around 480 seconds, and both DNN and DCN exceed 400 seconds. At inference, PrivateDFL achieves 4.19 milliseconds, while TabTransformer requires 344–346 milliseconds, DNN requires 428–471 milliseconds, and TabNet requires 68–88 milliseconds.

These timing results complement the accuracy analysis and motivate the energy evaluation that follows, where we examine whether the same advantages persist under power constraints.

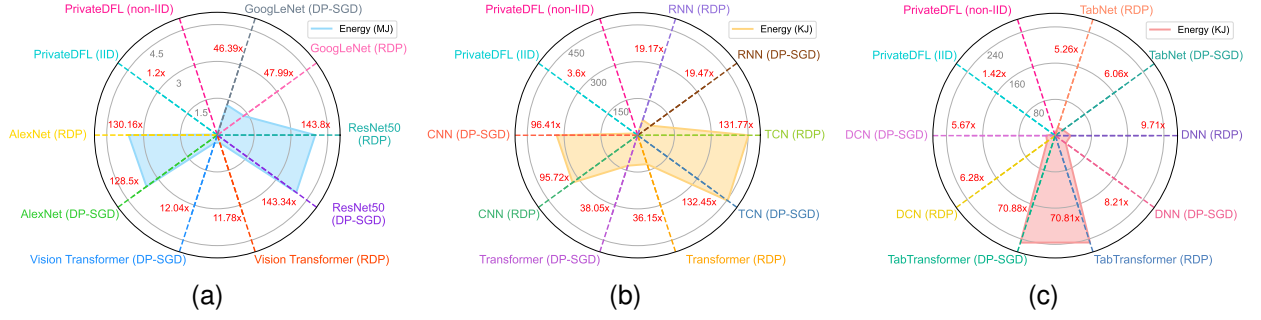


Figure 12: Energy consumption comparison between PrivateDFL and differentially private baseline models across the three benchmark datasets. Subfigures (a)–(c) correspond to MNIST, ISOLET, and UCI-HAR, respectively.

Building on the runtime analysis, Figure 12 reports the energy consumption of PrivateDFL and differentially private baseline models on MNIST, ISOLET, and UCI-HAR under IID and non-IID partitions. The trends closely follow the accuracy and timing results: PrivateDFL consistently meets the same privacy budgets with substantially lower energy overhead.

For MNIST (Figure 12a), PrivateDFL consumes only 0.03 MJ for both IID and non-IID settings. In contrast, DP deep learning baselines require significantly more energy: ResNet50 consumes 3.97–3.99 MJ, AlexNet about 3.6 MJ, GoogLeNet 1.29–1.33 MJ, and Vision Transformer 0.33 MJ. Thus, PrivateDFL achieves comparable image accuracy while using at least 11 \times less energy than the most efficient deep baseline (Vision Transformer) and up to roughly 130 \times less than ResNet50.

For ISOLET (Figure 12b), PrivateDFL consumes 12.32 kJ for IID and 3.43 kJ for non-IID. CNN models require around 330 kJ, TCN exceeds 450 kJ, and Transformer models range from 124 to 130 kJ. Even RNNs require 65–67 kJ. Consequently, PrivateDFL provides strong speech-recognition performance while using at least 5 \times less energy than the most efficient deep baseline (RNN) and up to around 130 \times less than TCN and CNN, with even larger savings under non-IID settings.

For UCI-HAR (Figure 12c), PrivateDFL consumes 4.96 kJ for IID and 3.50 kJ for non-IID. TabTransformer requires approximately 248 kJ, TabNet requires 18–21 kJ, and DNN and DCN models consume 20–34 kJ. Among all evaluated models, PrivateDFL is therefore the most energy-efficient solution for sensor-based recognition tasks.

Overall, the energy results reinforce the accuracy and runtime findings. PrivateDFL consistently achieves high utility while substantially lowering energy consumption across image, speech, and sensor modalities, underscoring its suitability for deployment in resource-constrained environments.

7 Conclusions and Future Work

This work presented PrivateDFL, a privacy-preserving decentralized learning framework that integrates HD with differentially private noise scheduling and transparent accounting. The framework provides three core advantages. First, HD classification offers an interpretable structure based on class prototypes and explicit update rules. Second, privacy is formally guaranteed through calibrated Gaussian noise whose magnitude is analytically derived for every client and communication round. Third, the lightweight HD operations make PrivateDFL practical on resource-constrained devices due to low computational cost and minimal communication overhead.

Although baseline models are trained centrally and therefore operate under more favorable conditions, PrivateDFL consistently matches or exceeds their performance under equal privacy budgets. Across image, speech, and sensor modalities, the framework achieves 95.74% on MNIST, 88.45% on ISOLET, and 94.30% on UCI-HAR under IID settings, with similar patterns under non-IID. The accuracy gains are substantial. Relative to Transformer models, the advantage reaches as high as 83% on ISOLET and about 31% on MNIST. Relative to deep neural networks, the advantage reaches 63% on ISOLET and 16% on MNIST, while on UCI-HAR PrivateDFL exceeds the strongest tabular baseline by about 15%. These improvements appear together with much lower training time, inference latency, and energy usage, which makes PrivateDFL an effective and efficient option for decentralized learning.

The characteristics of PrivateDFL directly support applications in domains where data cannot be centralized because of confidentiality, regulation, or ownership. Many real-world deployments involve sensitive information, including

patient health records, financial transactions, and government-managed personal identifiers. In such environments, strong privacy guarantees and transparent accounting are required for any collaborative learning system. PrivateDFL ensures that each participant’s contribution remains indistinguishable while tracking cumulative noise explicitly across all rounds, which avoids the severe utility loss that affects deep models trained with DP. Industrial environments, such as aerospace, automotive, and energy systems, also involve proprietary operational data. These settings require secure collaborative learning for tasks such as real-time defect detection, anomaly monitoring, and predictive maintenance. PrivateDFL is well-suited for these scenarios, since it maintains high accuracy under strict privacy constraints while operating efficiently on distributed devices.

Several limitations suggest directions for future research. The current framework assumes synchronous ring-style communication and reliable message passing. Extending the noise accounting to asynchronous schedules, dynamic topologies, and intermittent participation is an important next step. PrivateDFL also uses a uniform privacy target for all clients, and future work could consider adaptive privacy budgets that reflect local data difficulty or sensitivity while preserving global guarantees. Robustness to adversarial behavior requires deeper study, including targeted data poisoning, backdoor insertion, and strong inference attacks under heterogeneous participation. Finally, integrating secure aggregation, lightweight HE, or verifiable update mechanisms may further strengthen the protocol while maintaining its computational efficiency.

In summary, PrivateDFL shows that interpretable hyperdimensional representations, decentralized learning, and explainable DP can be combined to deliver high accuracy, strong privacy, low latency, and low energy consumption. These results support the use of PrivateDFL as a practical and scalable solution for trustworthy privacy-preserving learning in distributed and sensitive environments.

Acknowledgments

This work was supported by the National Science Foundation, United States [grant number 2434519].

Supplementary Materials

The code used in this study is openly available for reproducibility and can be accessed at the following link: <https://github.com/FardinJalilPiran/PrivateDFL.git>.

References

- [1] Mirsaeid Hosseini Shirvani and Mohammad Masdari. A survey study on trust-based security in internet of things: Challenges and issues. *Internet of Things*, 21:100640, 2023.
- [2] Betshrine Rachel Jibinsingh and Leema Nelson. FI-wosp: Federated learning with walrus optimization for sepsis prediction using mimic-iii physiological and clinical data. *Pattern Recognition*, page 112813, 2025.
- [3] Nuria Rodríguez-Barroso, Daniel Jiménez-López, M Victoria Luzón, Francisco Herrera, and Eugenio Martínez-Cámara. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. *Information Fusion*, 90:148–173, 2023.
- [4] Liangqi Yuan, Ziran Wang, Lichao Sun, S Yu Philip, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 2024.
- [5] Baosheng Li, Weifeng Gao, Jin Xie, Hong Li, and Maoguo Gong. Robust decentralized federated learning for heterogeneous and non-ideal networks. *Pattern Recognition*, 162:111362, 2025.
- [6] Chao Liu, Boxi Chen, Wei Shao, Chris Zhang, Kelvin KL Wong, and Yi Zhang. Unraveling attacks to machine learning-based iot systems: A survey and the open libraries behind them. *IEEE Internet of Things Journal*, 2024.
- [7] Chengcheng Wang, Xipeng P Tan, Shu Beng Tor, and CS Lim. Machine learning in additive manufacturing: State-of-the-art and perspectives. *Additive Manufacturing*, 36:101538, 2020.
- [8] Sweta Bhattacharya, Siva Rama Krishnan Somayaji, Thippa Reddy Gadekallu, Mamoun Alazab, and Praveen Kumar Reddy Maddikunta. A review on deep learning for future smart cities. *Internet Technology Letters*, 5(1):e187, 2022.
- [9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

- [10] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.
- [11] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345, 2017.
- [12] Xian Qin, Xue Yang, and Xiaohu Tang. Practical privacy-preserving federated learning based on multiparty homomorphic encryption for large-scale models. *Pattern Recognition*, page 112174, 2025.
- [13] Qiang Zhu and Xixiang Lv. 2p-dnn: Privacy-preserving deep neural networks based on homomorphic cryptosystem. *arXiv preprint arXiv:1807.08459*, 2018.
- [14] D Apple. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8):71, 2017.
- [15] Xiaofeng Ding, Cui Wang, Kim-Kwang Raymond Choo, and Hai Jin. A novel privacy preserving framework for large scale graph data publishing. *IEEE transactions on knowledge and data engineering*, 33(2):331–343, 2019.
- [16] Úlfar Erlingsson, Vasyi Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.
- [17] Junyoung Byun, Yujin Choi, and Jaewook Lee. Improving the utility of differentially private clustering through dynamical processing. *Pattern Recognition*, 157:110890, 2025.
- [18] Edwige Cyffers and Aurélien Bellet. Privacy amplification by decentralization. In *International Conference on Artificial Intelligence and Statistics*, pages 5334–5353. PMLR, 2022.
- [19] Muah Kim, Onur Günlü, and Rafael F Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2650–2654. IEEE, 2021.
- [20] Hanchi Ren, Jingjing Deng, and Xianghua Xie. Grnn: generative regression neural network—a data leakage attack for federated learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–24, 2022.
- [21] Nazish Khalid, Adnan Qayyum, Muhammad Bilal, Ala Al-Fuqaha, and Junaid Qadir. Privacy-preserving artificial intelligence in healthcare: Techniques and applications. *Computers in Biology and Medicine*, 158:106848, 2023.
- [22] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.
- [23] Wenzhi Fang, Yuning Jiang, Yuanming Shi, Yong Zhou, Wei Chen, and Khaled B Letaief. Over-the-air computation via reconfigurable intelligent surface. *IEEE transactions on communications*, 69(12):8612–8626, 2021.
- [24] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2021.
- [25] Chuan Ma, Jun Li, Ming Ding, Howard H Yang, Feng Shu, Tony QS Quek, and H Vincent Poor. On safeguarding privacy and security in the framework of federated learning. *IEEE network*, 34(4):242–248, 2020.
- [26] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR, 2020.
- [27] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer security—ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25*, pages 480–501. Springer, 2020.
- [28] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Third workshop on bayesian deep learning (NeurIPS)*, volume 12, 2018.
- [29] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.
- [30] Youyang Qu, Longxiang Gao, Tom H Luan, Yong Xiang, Shui Yu, Bai Li, and Gavin Zheng. Decentralized privacy using blockchain-enabled federated learning in fog computing. *IEEE Internet of Things Journal*, 7(6):5171–5183, 2020.
- [31] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

- [32] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.
- [33] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.
- [34] Maoguo Gong, Jialun Feng, and Yu Xie. Privacy-enhanced multi-party deep learning. *Neural Networks*, 121:484–496, 2020.
- [35] Huiwen Wu, Cen Chen, and Li Wang. A theoretical perspective on differentially private federated multi-task learning. *arXiv preprint arXiv:2011.07179*, 2020.
- [36] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380, 2022.
- [37] Anh-Tu Tran, The-Dung Luong, Jessada Karnjana, and Van-Nam Huynh. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422:245–262, 2021.
- [38] Bin Zhao, Kai Fan, Kan Yang, Zilong Wang, Hui Li, and Yintang Yang. Anonymous and privacy-preserving federated learning with industrial big data. *IEEE Transactions on Industrial Informatics*, 17(9):6314–6323, 2021.
- [39] Lihua Yin, Jiyuan Feng, Hao Xun, Zhe Sun, and Xiaochun Cheng. A privacy-preserving federated learning for multiparty data sharing in social iots. *IEEE Transactions on Network Science and Engineering*, 8(3):2706–2718, 2021.
- [40] Tianchong Gao, Hailong Fu, Shunwei Wang, and Niu Zhang. Amoue: Adaptive modified optimized unary encoding method for local differential privacy data preservation. *Computers and Electrical Engineering*, 120:109791, 2024.
- [41] Santanu Basak, Kakali Chatterjee, and Ashish Singh. Dppt: A differential privacy preservation technique for cyber-physical system. *Computers and Electrical Engineering*, 109:108661, 2023.
- [42] Waqas Ali, Mohammad Nauman, and Nouman Azam. A privacy enhancing model for internet of things using three-way decisions and differential privacy. *Computers and Electrical Engineering*, 100:107894, 2022.
- [43] Guiping Zheng, Bei Gong, Chong Guo, Tianqi Peng, and Mowei Gong. Awe-dpfl: Adaptive weighting and dynamic privacy budget federated learning for heterogeneous data in iot. *Computers and Electrical Engineering*, 123:110070, 2025.
- [44] Alejandro Hernández-Cano, Rosario Cammarota, and Mohsen Imani. Prid: Model inversion privacy attacks in hyperdimensional learning systems. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 553–558. IEEE, 2021.
- [45] Behnam Khaleghi, Mohsen Imani, and Tajana Rosing. Prive-hd: Privacy-preserved hyperdimensional computing. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
- [46] Fardin Jalil Piran, Prathyush P Poduval, Hamza Errahmouni Barkam, Mohsen Imani, and Farhad Imani. Explainable differential privacy-hyperdimensional computing for balancing privacy and transparency in additive manufacturing monitoring. *Engineering Applications of Artificial Intelligence*, 147:110282, 2025.
- [47] Fardin Jalil Piran, Zhiling Chen, Mohsen Imani, and Farhad Imani. Privacy-preserving federated learning with differentially private hyperdimensional computing. *Computers and Electrical Engineering*, 123:110261, 2025.
- [48] Fardin Jalil Piran, Anandkumar Patel, Rajiv Malhotra, and Farhad Imani. Domain-aware hyperdimensional computing for edge smart manufacturing. *arXiv preprint arXiv:2509.26131*, 2025.
- [49] Fardin Jalil Piran, Hamza Errahmouni Barkam, Mohsen Imani, and Farhad Imani. Hyperdimensional cognitive computing for lightweight cyberattack detection in industrial internet of things. In *Proceedings of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 7 of *19th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, page V007T07A013. American Society of Mechanical Engineers, 2023.
- [50] Zhiling Chen, Danny Hoang, Fardin Jalil Piran, Ruimin Chen, and Farhad Imani. Federated hyperdimensional computing for hierarchical and distributed quality monitoring in smart manufacturing. *Internet of Things*, 31:101568, 2025.
- [51] Fardin Jalil Piran, Prathyush P Poduval, Hamza Errahmouni Barkam, Mohsen Imani, and Farhad Imani. Privacy-preserving in-situ monitoring in additive manufacturing through hyperdimensional computing. In *ASME International Mechanical Engineering Congress and Exposition*, volume 88605, page V002T03A078. American Society of Mechanical Engineers, 2024.

- [52] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.